Simulador de un cluster tolerante a fallos basado en OMNeT++

C. Marcelo Pérez Ibarra, Cecilia M. Lasserre, Héctor P. Liberatori, Nilda M. Pérez Otero, Luis M. Valdiviezo[1]

[1]Facultad de Ingeniería - Universidad Nacional de Jujuy (UNJu) - Argentina

{cmperezi, classerre, hliberatori, nilperez, mlvaldiviezo}@fi.unju.edu.ar

Resumen

La Computación de Altas Prestaciones mediante *clusters* de computadoras basados en *workstation* y redes, posibilitó la construcción y uso de computadoras paralelas. Su principal objetivo es el aumento de prestaciones, utilizando el potencial ofrecido por un gran número de procesadores. La construcción de un *cluster* tiene tres retos: **Alto Rendimiento**, **Alta Disponibilidad** y **Alta Productividad**. Respecto a la Alta Disponibilidad se debe considerar la probabilidad de fallos o desconexión de nodos por lo que se debe reducir el tiempo medio entre fallos. Para ello es necesario definir la configuración adecuada de tolerancia a fallos para diferentes tipos de aplicaciones, teniendo en cuenta los requerimientos de rendimiento y prestaciones del usuario. Para definir y validar un modelo genérico de aplicación-prestación-tolerancia a fallos es necesario realizar mediciones en un *cluster*. La realización de estas mediciones, considerando diferentes configuraciones de *cluster*, tipos de aplicaciones y la tolerancia a fallos, resulta compleja. Por ello, en este proyecto se pretende desarrollar un simulador para la arquitectura tolerante a fallos RADIC. El entorno sobre en cual se desarrollará el simulador será OMNeT++ junto con el framework INET.

Palabras clave: Computación de Altas Prestaciones, HPC, Alta Disponibilidad, Tolerancia a Fallos, RADIC, OMNeT++.

Contexto

La línea actual de investigación está inserta en el proyecto ***Sistemas de Cómputo de Altas Prestaciones con Alta Disponibilidad: Evaluación de la Performance en Diferentes Configuraciones***. En este proyecto trabajan, además de los autores de este artículo, Sandra Méndez y Rodrigo Cachambe. El proyecto está acreditado y financiado por la Secretaria de Ciencia y Técnica y Estudios Regionales de la Universidad Nacional de Jujuy (SECTER-UNJu) y se inserta en el programa de incentivos al docente-investigador bajo el código 08-D0093. El proyecto está asesorado por los doctores Emilio Luque Fadón y Dolores Isabel Rexachs del Rosario, pertenecientes a la Escuela Técnica Superior de Ingeniería, Dpto. Arquitectura de Computadoras y Sistemas Operativos de la Universidad Autónoma de Barcelona.

1. Introducción

La llegada de la Computación de Altas Prestaciones (*HPC - High Performance Computing*) a través de *clusters* de computadores basados en *workstation* (WS) y redes convencionales (local o remota) o COTS (*Commodity Off The Shelf*), ha posibilitado que muchas instituciones y organizaciones vean viable la construcción y el uso de computadores paralelos. Los sistemas de Cómputo de Altas Prestaciones se utilizan para desarrollar software en una gran variedad de dominios, incluyendo física nuclear, simulación de accidentes, procesamiento de datos de satélites, dinámica de fluidos, modelado del clima, bioinformática y modelado financiero. El sitio web TOP500 [1] lista los 500 mejores sistemas de cómputo de altas prestaciones. La diversidad de organizaciones científicas, gubernamentales y comerciales presentes en esta lista ilustra el creciente predominio e impacto de las aplicaciones de HPC en la sociedad moderna [1]. En estas arquitecturas paralelas el objetivo principal es el aumento de la capacidad de procesamiento, utilizando el potencial ofrecido por un gran número de procesadores (alto aumento del *speedup* y alta disponibilidad) [2]. La construcción de un *cluster* tiene tres retos: Alto Rendimiento (*High Performance: HP*), Alta Disponibilidad (*High Availability: HA*) y Alta Productividad (*High Throughput: HT*) [3] [4]. Manejar eficientemente un número elevado de computadores en ambientes heterogéneos

[1]http://www.top500.org

(heterogeneidad de nodos de cómputo, de redes, etc.) no es trivial, por ello requiere un cuidadoso diseño de la arquitectura y su funcionalidad. Esta arquitectura debe garantizar que se realice la ejecución eficiente de una aplicación paralela cuando se utiliza un entorno con un elevado número de nodos, durante un largo período de tiempo. Por otro lado, es necesario considerar que la probabilidad de fallo aumenta, y puede llegar a ocurrir que el fallo ocasione la pérdida total del trabajo realizado. Un *cluster* de *HA* intenta mantener en todo momento la prestación de servicios encubriendo los fallos que se puedan producir. Por lo tanto para proveer *HA* es necesario utilizar técnicas de tolerancia a fallos [5] que permitan garantizar (confiabilidad o garantía de servicio) Fiabilidad (*Reliability*), Disponibilidad (*Availability*) y Facilidad de Mantenimiento (*Serviceability*) del sistema. Para evitar la interrupción en el suministro del servicio, debido a algún fallo en sus componentes, los fallos deben ser detectados lo más rápidamente posible (latencia del fallo). El nodo en que ha ocurrido el fallo debe ser identificado a través de un diagnóstico apropiado y finalmente reparado o aislado a través de la reconfiguración del sistema. Esta reconfiguración vuelve a asignar las tareas y selecciona caminos alternativos de comunicación entre los nodos. Los fallos en *clusters* pueden ser causados por problemas en el hardware, sistema operativo (OS), aplicaciones, o por el propio software de tolerancia a fallos [6]. Evidentemente es necesario utilizar alguna técnica o mecanismo que proteja al sistema, esto se lleva a cabo mediante la prevención (generar redundancia), la detección (monitorización), el diagnóstico (latencia del error), la recuperación y la reconfiguración del sistema (retornar el sistema a condiciones operativas razonables después un fallo) como un todo. Para lograr todo esto, la tolerancia a fallos incluye las siguientes etapas:

- Protección: redundancia.
- Detección del error.
- Estimación de daños: diagnosis.
- Recuperación del error: llevar al sistema a un estado correcto, desde el que pueda seguir funcionando (tal vez con funcionalidad parcial).
- Tratamiento del fallo y continuación del servicio del sistema, consistencia global del sistema, reconfiguración y enmascaramiento del fallo.

Debe mantenerse un equilibrio entre la latencia y el *overhead* que introducen las técnicas de tolerancia a fallos y considerar el impacto del incremento del coste del sistema (desde el punto de vista del usuario) y la reducción de prestaciones asociada a las actividades realizadas por el mecanismo de tolerancia a fallos. Considerando estos aspectos, Duarte *et al.* han propuesto y desarrollado RADIC (**R**edundant **A**rray of **D**istributed **I**ndependent Fault Tolerance **C**ontrollers) [7]. RADIC [8] es una arquitectura tolerante a fallos escalable, flexible, transparente y descentralizada para sistemas de paso de mensajes, cuyo objetivo principal es asegurar que una aplicación paralela-distribuida finalice correctamente aún si ocurren fallos en algunos nodos de la computadora paralela. RADIC establece un modelo de arquitectura que define la interacción de la arquitectura tolerante a fallos y la estructura de la computadora paralela, implementando una capa entre el nivel de paso de mensajes y la estructura de la computadora. Las fases de RADIC se suceden concurrentemente con la ejecución de la aplicación paralela y no interfiere en sus resultados [8]. Actualmente, es cada vez más frecuente el uso de modelos de simulación computacional en HPC, ya sea como ayuda modelado de prestaciones [9], como para explorar arquitecturas o aplicaciones [10] [11] o como una herramienta de de predicción de tráfico [12]. De igual modo, al realizar un simulador de RADIC se podrá disponer de una herramienta para analizar las características tolerantes a fallos de RADIC tales como el número ideal de nodos spare y su ubicación ideal dentro del *cluster*, investigar si es posible alcanzar mejores resultados distribuyendo los nodos spare de acuerdo con ciertos criterios (nivel de degradación aceptable, límites de memoria de un nodo, topología de red), investigar acerca de las posibles políticas a usarse en las tareas de reemplazo de nodo, etc. El proyecto original en el que está inserta la presente línea de investigación, ***Sistemas de Cómputo de Altas***

Prestaciones con Alta Disponibilidad: Evaluación de la Performance en Diferentes Configuraciones, pretende:

1. Definir la configuración adecuada de tolerancia a fallos para diferentes tipos de aplicaciones, teniendo en cuenta los requerimientos de rendimiento y prestaciones del usuario.
2. Definir un modelo genérico de aplicación-prestación-tolerancia a fallos.
3. Validar experimentalmente el modelo propuesto.

La metodología de este proyecto consta de los siguientes pasos:

- Recopilación, selección y análisis de la bibliografía existente sobre la teoría general de tolerancia a fallos, analizando las posibilidades de aplicación en el ámbito de redes y para sistemas de cómputo de altas prestaciones.
- Selección de un conjunto de aplicaciones que se ejecutarán sin tolerancia a fallos y con la arquitectura tolerante a fallos RADIC para diferentes niveles de protección e inyección de fallos.
- Búsqueda, evaluación y selección de un sistema de simulación que permita diseñar e implementar diferentes configuraciones de computadoras paralelas con diferentes números de nodos de cómputo y topología de red, para la experimentación.
- Validación de los resultados obtenidos.

En el contexto del tercer paso es que se instrumenta el presente proyecto: ***Simulador de un Cluster Tolerante a Fallos basado en OMNeT++***. OMNeT++ [2] es un entorno de desarrollo de simulación modular de eventos discretos de redes orientado a objetos, usado habitualmente para modelar el tráfico de redes de telecomunicaciones, protocolos, sistemas multiprocesadores y distribuidos, validación de arquitecturas hardware, evaluación del rendimiento de sistemas software y, en general, modelar cualquier sistema que pueda simularse con eventos discretos. Con el objetivo de desarrollar un simulador que permita experimentar con diferentes configuraciones, tipos de aplicación y mecanismos de tolerancia a fallos, se ha estudiado en profundidad el entorno de desarrollo de OMNeT++, y actualmente se están realizando pruebas para llevar a cabo la simulación de aplicaciones paralelas. El principal inconveniente de realizar pruebas simuladas es la exactitud de los resultados. Las simulaciones deben obtener resultados que sean lo suficientemente similares a aquellos obtenidos en un sistema real. Obtener y asegurar esa precisión es el problema que debe ser resuelto cuando se diseñan nuevos entornos de simulación [13]. Es por ellos que con el fin de conseguir simulaciones fieles a la realidad, las cargas del simulador se generarán a partir de trazas de aplicaciones reales, obtenidas sobre un *cluster* sobre el que se ha instalado la librería de paso de mensajes MPICH. Junto con MPICH viene una librería (MPI Parallel Environment - MPE) que permitirá generar los archivos de trazas de aplicaciones paralelas.

2. Líneas de Investigación y Desarrollo

Las líneas de investigación son:

- Simulación de un *cluster* usando el entorno de simulación OMNeT++
- Simulación de RADIC usando el entorno de simulación OMNeT++

3. Resultados Obtenidos/Esperados

Resultados obtenidos

- Trazas de aplicaciones paralelas.

Resultados a obtener

- Simulación de un *cluster* tolerante a fallos usando el entorno de simulación OMNeT++.

4. Formación de Recursos Humanos

Equipo de trabajo de la línea de I/D presentada:
Directora: Nilda Pérez Otero
Co-Director: Marcelo Pérez Ibarra *Otros integrantes*:

- Cecilia Lasserre
- Héctor Liberatori
- Luis Valdiviezo

[2] http://www.omnetpp.org/

Tesis de Posgrado y Tesinas de Grado, en curso que se relacionan directamente con la línea de I/D presentada:

- Tesis de maestría en curso: 1
- Tesinas en curso: 2

Tesis de Posgrado y Tesinas de Grado en curso, que se relacionan directamente con la línea de I/D que involucra la presentada:

- Doctorado en curso en la UAB: 1
- Especialización en UNLP: 1
- Maestría en la UNSL: 1

Referencias

[1] Jeffrey C. Carver. Third international workshop on software engineering for high performance computing (hpc) applications. In *ICSE COMPANION '07: Companion to the proceedings of the 29th International Conference on Software Engineering*, page 147, Washington, DC, USA, 2007. IEEE Computer Society.

[2] Josemar Rodrigues de Souza. *FT-DR: Tolerancia a fallos, en clusters de computadores geográficamente distribuidos, basada en Replicación de Datos*. PhD thesis, Departament d'Arquitectura de Computadors i Sistemes Operatius. Universitat Autònoma de Barcelona, http://www.tdx.cat/TDX-1013106-133133, jun 2006.

[3] Rajkumar Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[4] Thomas Sterling. Launching into the future of commodity cluster computing. In *CLUSTER '02: Proceedings of the IEEE International Conference on Cluster Computing*, page 345, Washington, DC, USA, 2002. IEEE Computer Society.

[5] Algirdas Avizienis. Toward systematic design of fault-tolerant systems. *Computer*, 30(4):51–58, 1997.

[6] Hairong Sun, Jame J. Han, and Haim Levendel. A generic availability model for clustered computing systems. In *PRDC '01: Proceedings of the 2001 Pacific Rim International Symposium on Dependable Computing*, page 241, Washington, DC, USA, 2001. IEEE Computer Society.

[7] Angelo Duarte, Dolores Rexachs, and Emilio Luque. Increasing the cluster availability using radic. In *CLUSTER*, 2006.

[8] Angelo Duarte. *RADIC: A Powerful Fault-Tolerant Architecture*. PhD thesis, Departament d'Arquitectura de Computadors i Sistemes Operatius. Universitat Autònoma de Barcelona, http://www.tdx.cat/TDX-1126107-101303, may 2007.

[9] Wolfgang E. Denzel, Jian Li, Peter Walker, and Yuho Jin. A framework for end-to-end simulation of high-performance computing systems. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[10] S. D. Hammond, G. R. Mudalige, J. A. Smith, S. A. Jarvis, J. A. Herdman, and A. Vadgama. Warpp: a toolkit for simulating high-performance parallel scientific codes. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[11] Cyriel Minkenberg and Germán Rodriguez. Trace-driven co-simulation of high-performance computing systems using omnet++. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences,

Social-Informatics and Telecommunications Engineering).

[12] Mustafa M. Tikir, Michael A. Laurenzano, Laura Carrington, and Allan Snavely. Psins: An open source event tracer and execution simulator for mpi applications. In *Euro-Par '09: Proceedings of the 15th International Euro-Par Conference on Parallel Processing*, pages 135–148, Berlin, Heidelberg, 2009. Springer-Verlag.

[13] Alberto Nuñez, Javier Fernández, Jose D. Garcia, Laura Prada, and Jesús Carretero. Simcan: a simulator framework for computer architectures and storage networks. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).